

Lecture 11

Databases

Why Databases are useful

- Using flat files (ex. VSAM), each file is stored in a separate data set in sequential or indexed format.
- To retrieve data from the file, an application had to open the file and read through it to the location of the desired data.
- If the data was scattered through a large number of files, data access required a lot of opening and closing of files, creating additional I/O and processing overhead.
- To reduce the number of files accessed by an application, programmers often stored the same data in many files.
- This practice created redundant data and the related problems of ensuring update consistency across multiple files.
- To ensure data consistency, special cross-file update programs had to be scheduled following the original file update.

Data Duplication

Sequential & Indexed Files (ex. VSAM)

Data Duplication?

Name	Address	Course	Grade
Mr. Eric Tachibana	123 Kensington	Chemistry 102	C+
Mr. Eric Tachibana	123 Kensington	Chinese 3	A
Mr. Eric Tachibana	122 Kensington	Data Structures	B
Mr. Eric Tachibana	123 Kensington	English 101	A
Ms. Tonya Lippert	88 West 1st St.	Psychology 101	A
Mrs. Tonya Duc	100 Capitol Ln.	Psychology 102	A
Ms. Tonya Lippert	88 West 1st St.	Human Cultures	A
Ms. Tonya Lippert	88 West 1st St.	European Gov	A

Why DBMS's are useful (cont)

- A database stores the data only once in one place and makes it available to all application programs and users.
- Data can also be backed up and recovered more easily in a single database than in a collection of flat files.
- Databases provide security by limiting access to data. The ability to read, write, update, insert, or delete data can be restricted.
- Database structures offer multiple strategies for data retrieval (sequentially or go directly to the desired data).
- Finally, an update performed on part of the database is immediately available to other applications. Because the data exists in only one place, data integrity is more easily ensured.

Database

- A *Database* is a collection of interrelated data items, stored once and organized in a form for easy retrieval.
- The data (in the database) is often stored in VSAM files (but the VSAM files are not accessed directly by the Database user), but rather, through a Database Management System

Database Management Systems (DBMS)

- A Database Management System is a collection of programs for storing organizing, selecting, modifying, and extracting data from a database.
- A DBMS defines and maintains the structure of the database by providing support for a business application transaction or process to access stored information
- The two most common types of database structures are relational and hierarchical.

The role of a DBMS is to provide:

1. Enable concurrent access (I.e. multiple users, multiple applications) to a single copy of the data
2. Control concurrent access via a locking mechanism with **ACID properties** (ACID is an acronym for atomicity, consistency, isolation, and durability) so as to maintain integrity for all updates
3. Reduce data redundancy by maintaining only one copy of the data
4. Automated rollback, restart and recovery

ACID

- Atomicity - the transaction must execute completely, or not at all
- Consistency - Resources must remain consistent (ex. Same amount of money debited and credited)
- Isolation - each transaction must appear to occur before or after any other transaction
- Durability - If the transaction completes successfully, the state (changes due to the transaction) will survive any future failures

Role of the database administrator

Provides standards for databases; administers databases



Guides, reviews and approves database designs

Determines rules for accessing data and monitors its security

Controls database integrity & availability; monitors activities for backup and recover

Approves the use of any programs that access production databases

Why use a database/DBMS?

- Reduce programming effort
- Manage data more efficiently
- Easy to separate confidential/sensitive info
- Provide a greater level of security
- Access & update simultaneously
- Ensure consistency
- Provide backup and recovery
- Utilities to monitor and tune
- Structure change does not impact existing developments

Types of Databases on z/OS

“Flat Files”

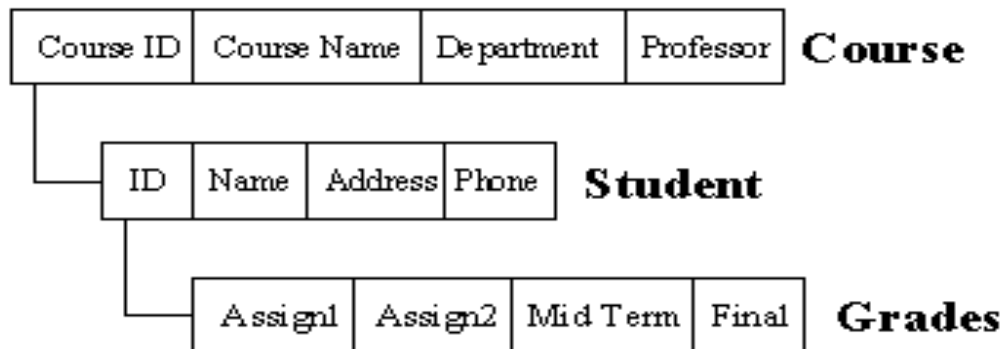
Sequential & Indexed Files (ex. VSAM)

Data Duplication?

Name	Address	Course	Grade
Mr. Eric Tachibana	123 Kensington	Chemistry 102	C+
Mr. Eric Tachibana	123 Kensington	Chinese 3	A
Mr. Eric Tachibana	122 Kensington	Data Structures	B
Mr. Eric Tachibana	123 Kensington	English 101	A
Ms. Tonya Lippert	88 West 1st St.	Psychology 101	A
Mrs. Tonya Duc	100 Capitol Ln.	Psychology 102	A
Ms. Tonya Lippert	88 West 1st St.	Human Cultures	A
Ms. Tonya Lippert	88 West 1st St.	European Gov	A

Hierarchical Storage

- Parent / Child Relationship
- Examples of Hierarchical exists all around (ex. Windows / Unix Directory Structure)
- Much Less Redundant Data

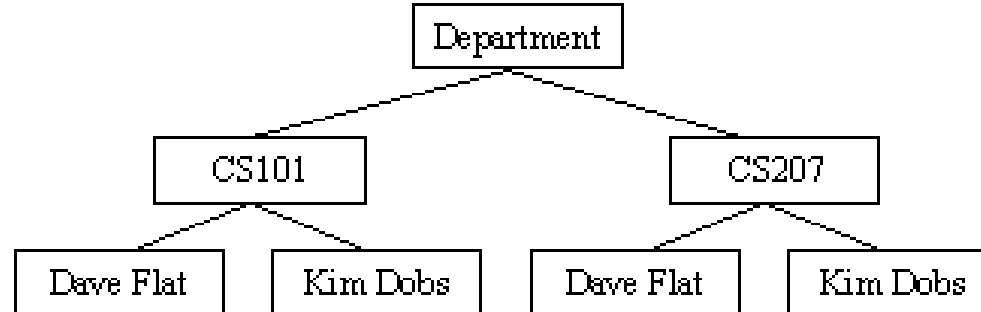


Hierarchical Storage

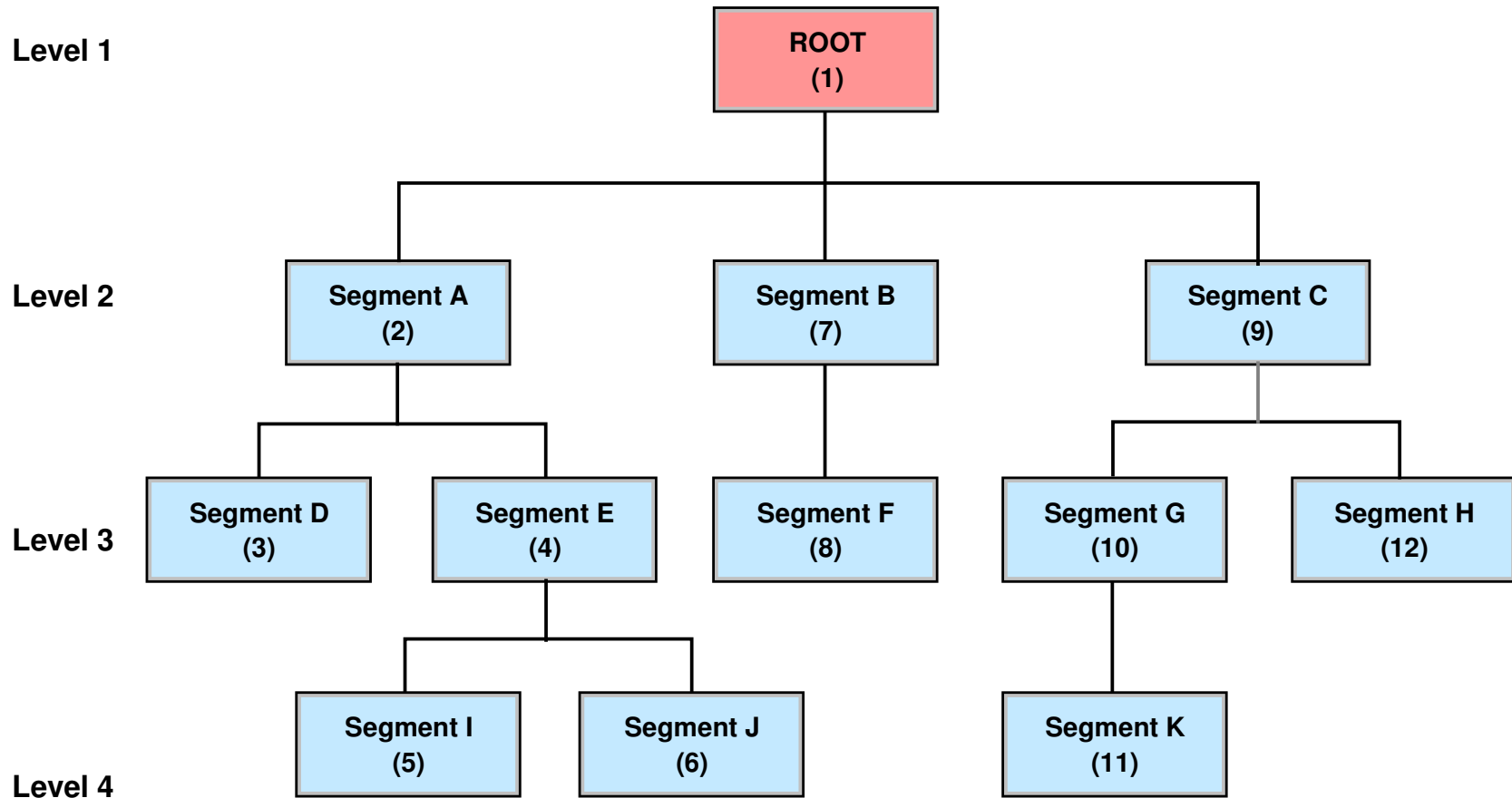
Is there Still Data Duplication?

Hierarchical Storage

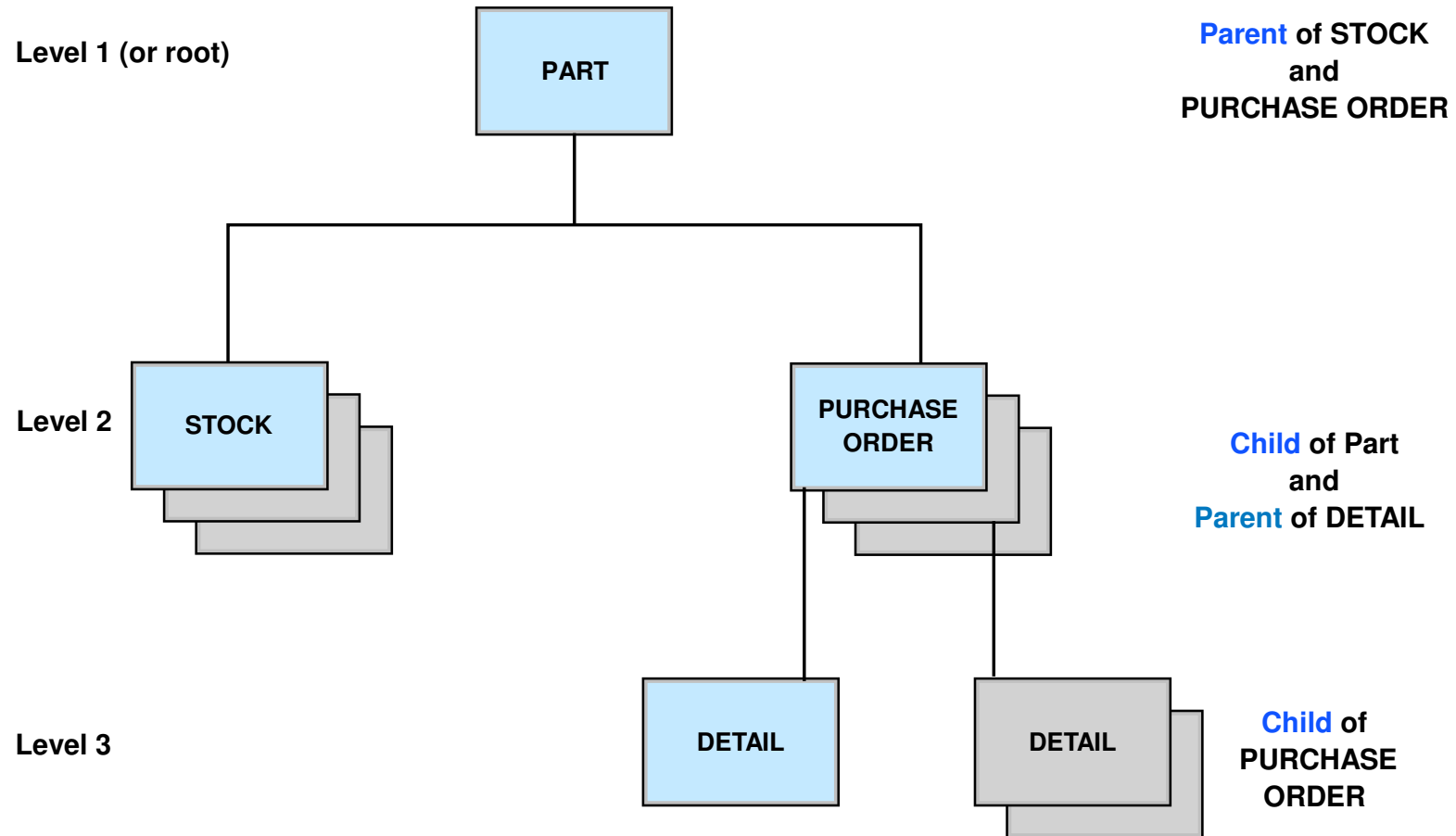
Is there Still Data Duplication?



Hierarchical DB : Relationships & sequence



Hierarchical data structure



Relational Databases

Relational DBMS: Overview

Key Relational DB Concepts:

- Tables (or relations) -- all data is stored within a table.
 - Each table is made up of:
 - Records (horizontal rows also known as tuples)
 - Fields (vertical columns also known as attributes)
 - » Can think of tables as either records or fields
 - » Each table has a unique name
- Access information via SQL (Structured Query Language)
 - Ex. Give me all rows where course = “CS201”

Relational DBMS: Sample Tables / Relations

Student Table

S_ID	LastN	FirstN	Address	City	State	Zip	Phone
01	Jones	Jane	10th St.	Chester	PA	55505	444-3333
02	Lim	Jason	Main St.	Oaks	PA	55505	333-2212
03	Wolf	Peter	Peach Rd	LA	CA	15505	121-5555

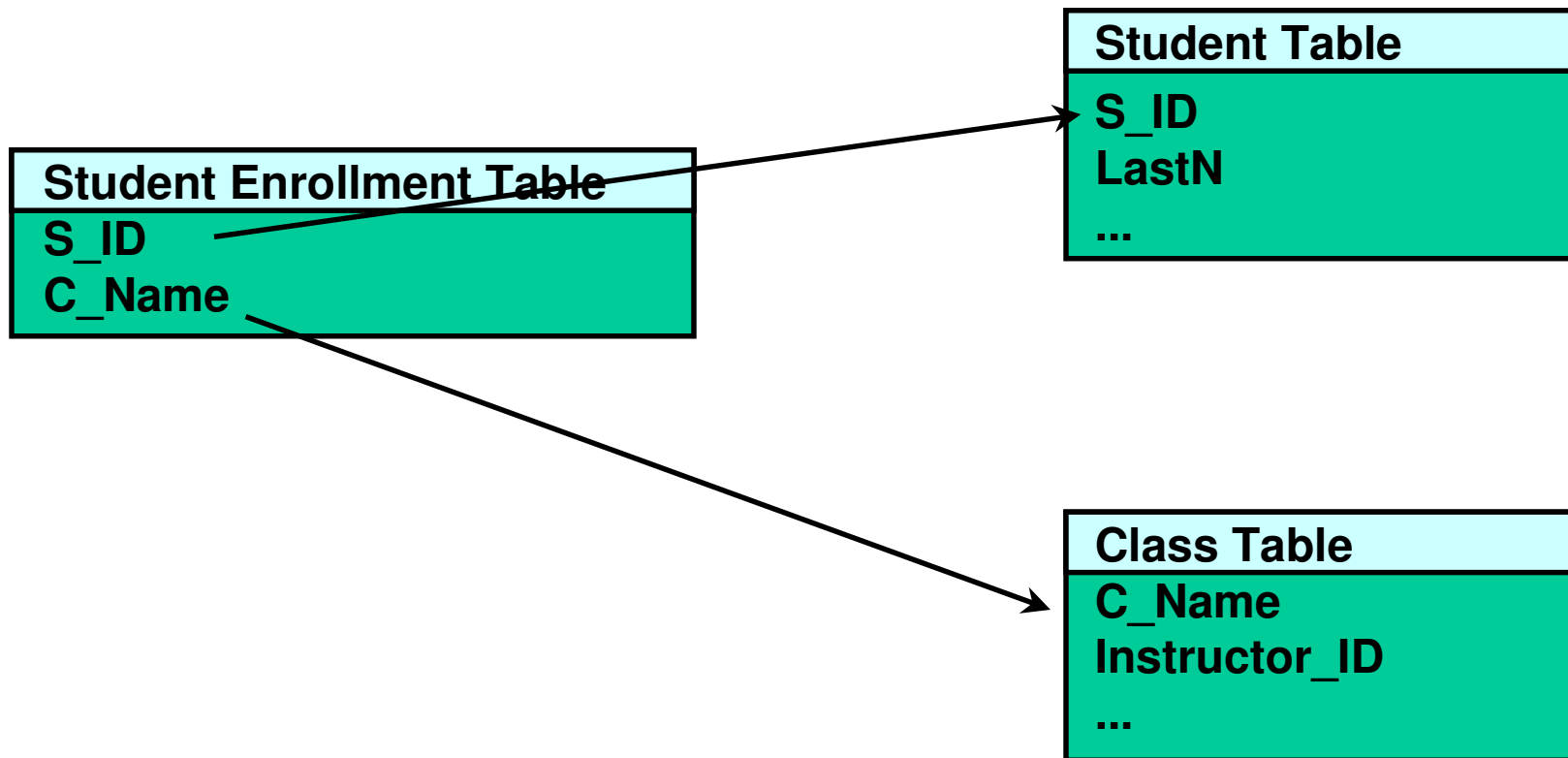
Class Table

C_Name	Instructor_ID	Num Credits
CS100	002	3
CS201	004	3
Eng351	003	4

Student Enrollment Table

S_ID	C_Name
02	CS100
03	CS100
03	Eng201

Relational Database Example: A 3 Table “Course” DB



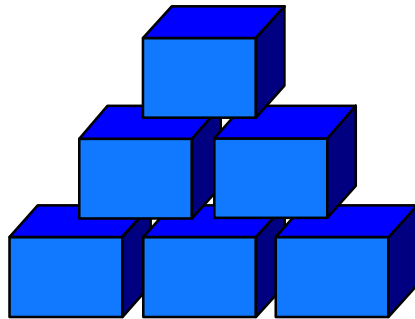
Comparing DB2 & IMS

Databases on z/OS

- IMS - Hierarchical database management system
- DB2 - Relational database management system

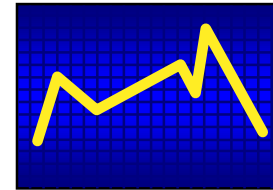
Data Characteristics

Operational Data



Application Oriented
Limited Integration
Constantly Updated
Current Values Only
Supports Daily Operations

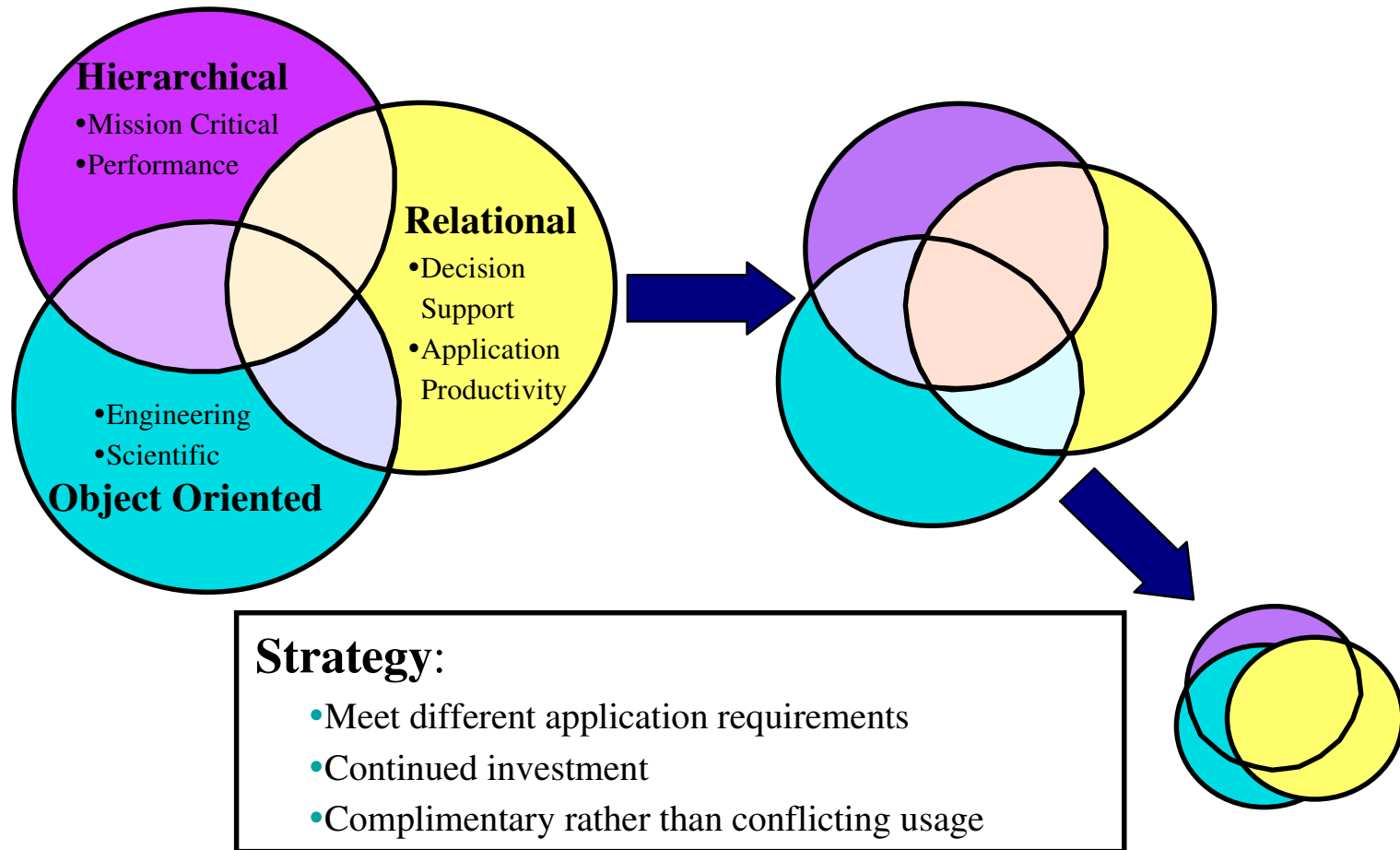
Analytical Data



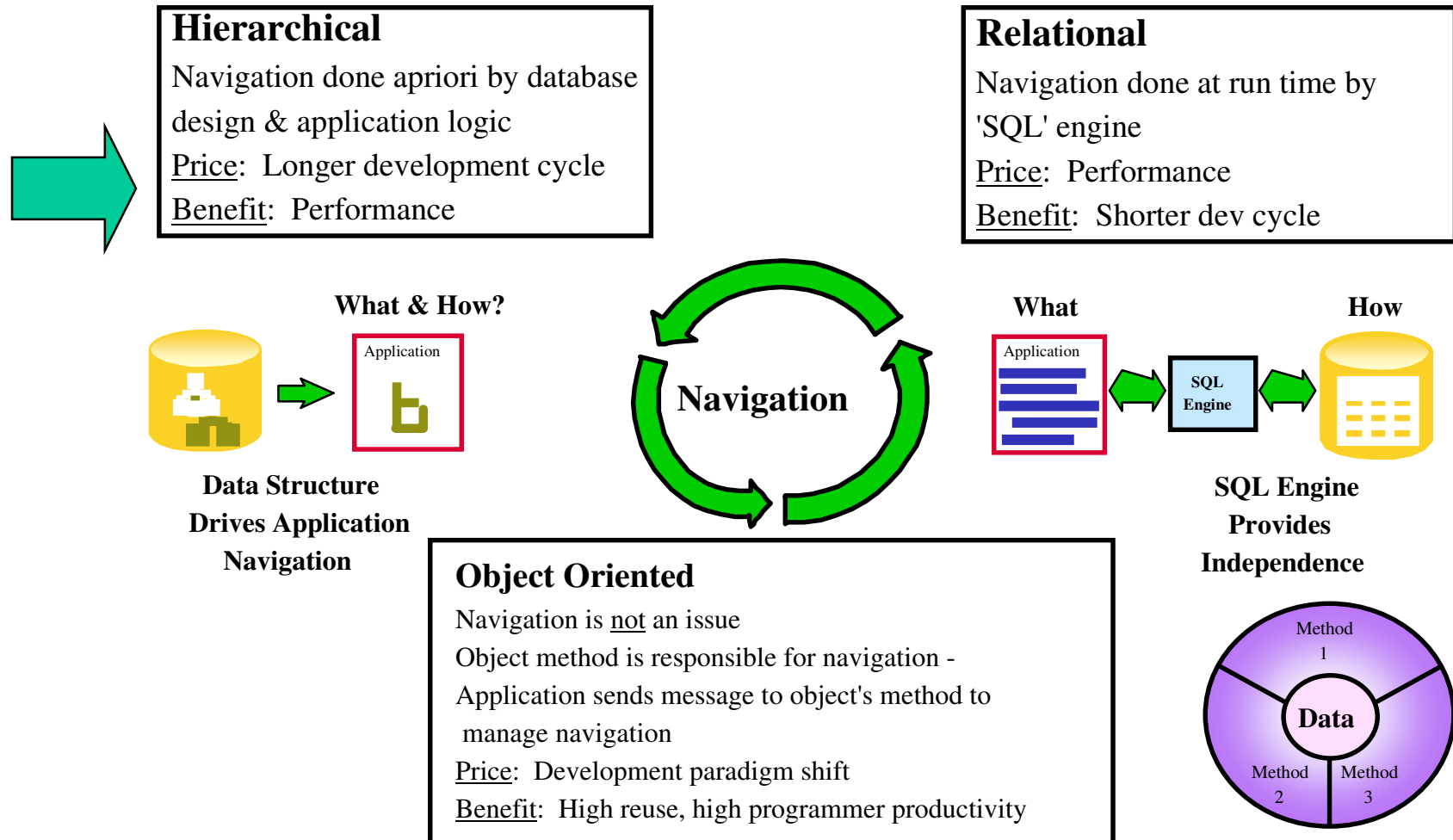
Subject Oriented
Integrated
Non-volatile
Values Over Time
Supports Decision Making

Operational and Analytical Data Fundamentally Different

Database Manager Positioning



Database Management Systems - Differentiation



A database comparison:

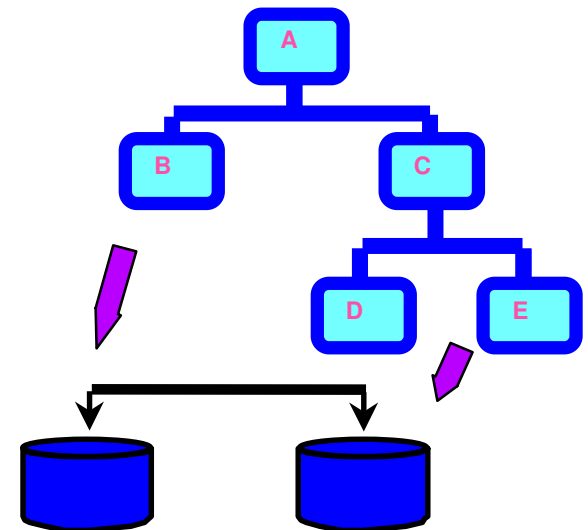
- **Hierarchical (IMS):**
 - Data is relatively static (can not just “add a column”)
 - Navigational : need to know the structure to get to the data
 - Faster & more efficient - once Segment Search Arguments (SSAs) are coded
- **Relational (DB2):**
 - Changeable info
 - Change in structure : no impact on existing application
 - Non-Navigational : no need to know the structure to get to the right data (just tablename and columnname(s))
- **Relational DBs provides well for unstructured or unplanned access to data, but typically has significantly higher processing cost and lower performance than a Hierarchical database**

IMS Hierarchical Database

IMS is a Hierarchical Database Management System

IMS DB is organized hierarchically

- To optimize storage and retrieval
- To ensure integrity and recovery



History of IMS

- IBM designed IMS with Rockwell and Caterpillar starting in 1966 for the Apollo program.
- IMS's challenge was to inventory the very large Bill of Materials for the Saturn V moon rocket and Apollo space vehicle.
- Reportedly IMS alone is a \$1 billion (U.S.) per year business for IBM

The World Depends on IMS

IMS is a part of everyday life. When you:

<i>Turn on a light</i>	<i>Get a business loan</i>
<i>Make a telephone call</i>	<i>Process accounting records</i>
<i>Use your ATM card</i>	<i>Control inventories</i>
<i>Put money in a bank</i>	<i>Process payroll</i>
<i>Rent a car</i>	<i>Update personnel records</i>
<i>Purchase life insurance</i>	<i>Control an assembly line</i>
<i>Travel</i>	<i>Control a railroad</i>
<i>Send a package</i>	<i>Use corporate data bases</i>
<i>Track in-transit packages</i>	<i>Run a government agency</i>
<i>Trade stocks</i>	<i>Conduct international business/banking</i>

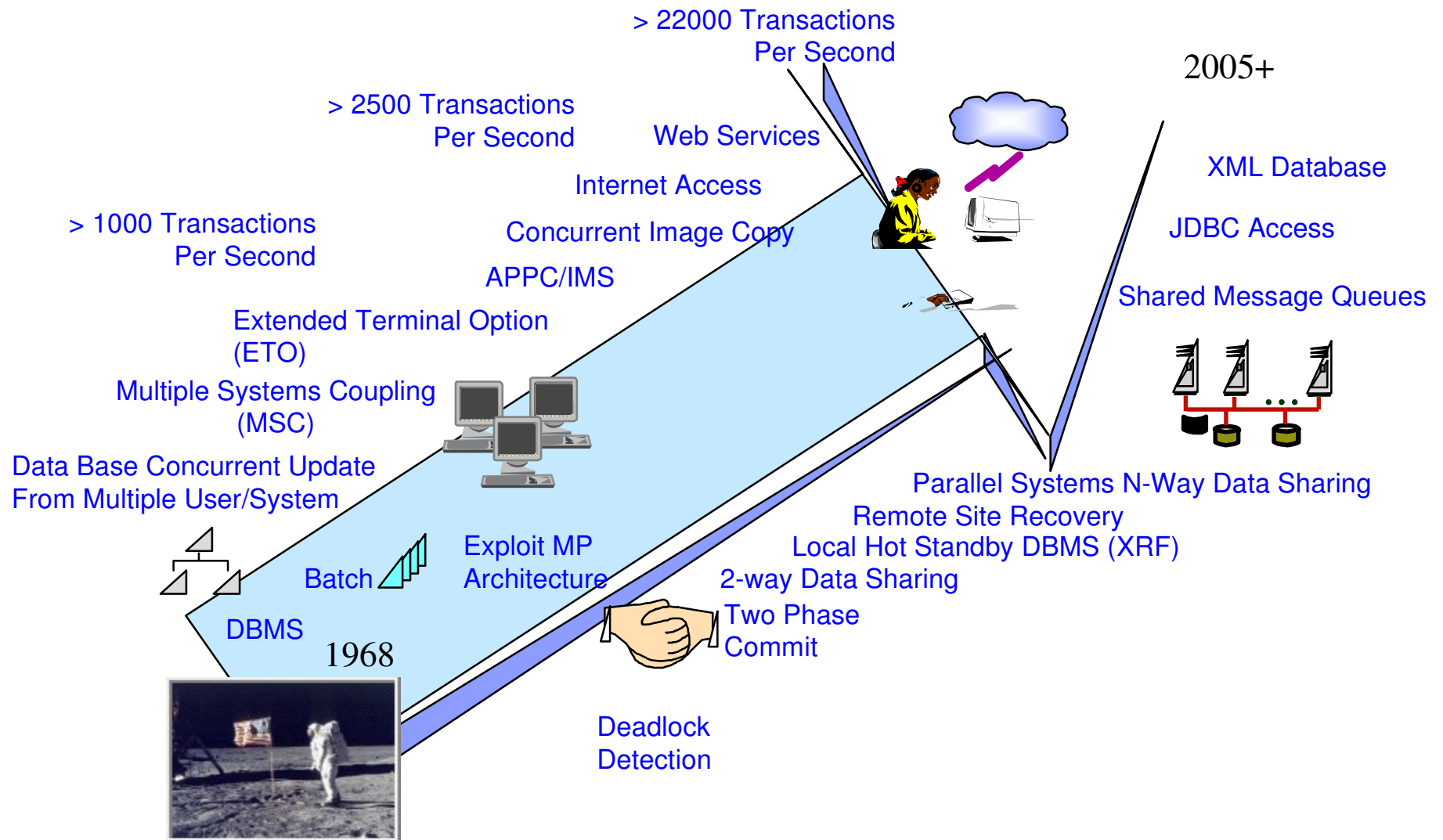
And more...

... you are likely using IMS!

An IMS Customer Then and Now

	An Original Environment in 1969	A Large Customer in 2005
Number of Terminals	139 on 110 Lines	Tens of Thousands
Number of Data Bases	30	Thousands
DASD used to hold Data Bases	4 - 2314 DASD	Terabytes of DASD
Number of Transactions per Day	17,000 - 20,000	Over 100 Million
Number of Transaction Codes	260	Thousands
Number of Applications	8	Thousands
System Availability	Less Than 24 Hours	3 Hours of Planned and Unplanned Outages per Year
Response Time	2-5 Seconds	Sub-Second

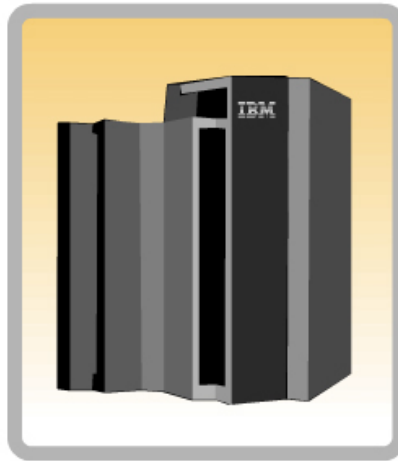
37+ Years



IMS and zSeries -- Scalability

**Benchmarked over 22,000 IMS Transactions/Second
with Database Update on a SINGLE SYSTEM**

Approximately 2 billion transactions/day



IMS V9 continues to leverage zSeries leadership capabilities offering a broad range of scalability and continually increasing performance/ capacity

Significantly more transaction throughput

Faster shared queues and shared data

Increased I/O bandwidth

Practically limitless volumes of data

IMS Runs the World...

Most Corporate Data is Managed by IMS

- **Over 95 % of top Fortune 1000 Companies use IMS**
- **IMS Manages over 15 Billion GBs of Production Data**
- **\$2.5 Trillion/day transferred through IMS by one customer**

Over 50 Billion Transactions a Day run through IMS

- **IMS Serves Close to 200 Million Users a Day**
- **Over 100 Million IMS Trans/Day Handled by One Customer on a single system**
- **120M IMS Trans/day, 7M per hour handled by another customer**
- **6000 Trans/sec across TCP/IP to single IMS with a single Connect instance**
- **Over 21,000 Trans/sec (near 2 Billion/day) with IMS Data/Queued sharing on a single processor**

Gartner Group: "A large and loyal IMS installed base. Rock-solid reputation of a transactional workhorse for very large workloads. Successfully proven in large, Web-based applications. IMS is still a viable, even unmatched, platform to implement very large OLTP systems, and, in combination with Web Application Server technology, it can be a foundation for a new generation of Web-based, high-workload applications."

IMS Today

(extra reading, not part of lecture)

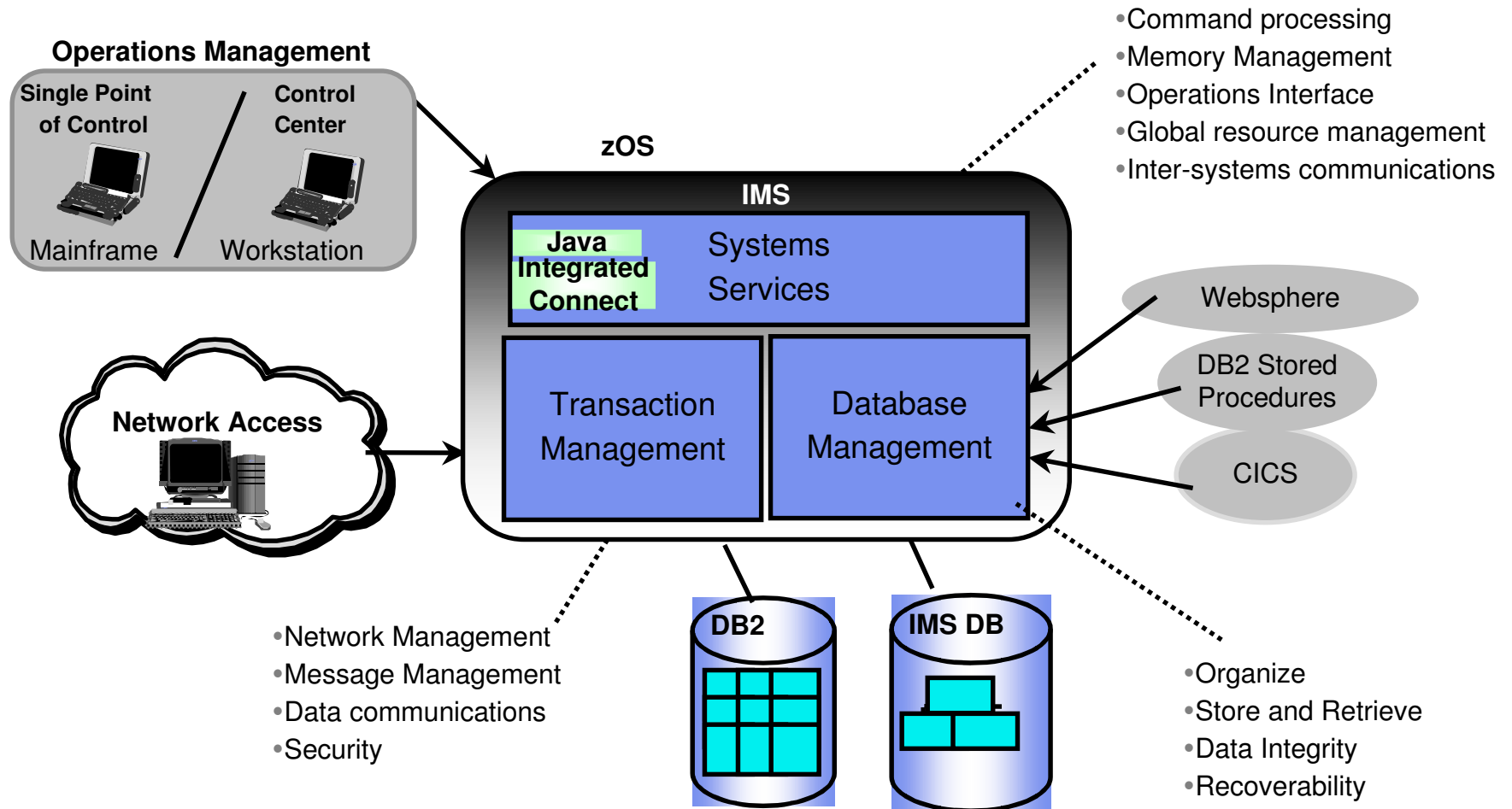
What is IMS?

Information Management System

3 components:

- The Transaction Manager (TM)
- The Database Manager (DB)
- Set of system services, providing common services to TM & DB

IMS Environment



IMS Transaction Manager (TM) messages

- An IMS application program runs “under the control” of either IMS Batch, IBM TM or CICS.
- This “controlling system” takes care of basic I/O operations (open, close, log, concurrency, recovery)
- Without the TM component, you would be able to process data in the IMS database in a batch mode only. With the IMS TM component, you can access the data and can perform update, delete, and insert functions online

IMS DB

Three basic forms of IMS hierarchical databases

1. "Full function" databases
2. "Fast path" databases
3. High Availability Large Databases

“Full Function” IMS databases

- Directly descended from the same Data Language/1 (DL/I) databases as developed for Apollo.
- Can have primary and secondary indexes
- Are accessed using DL/I calls from your application program, not all that unlike SQL calls to DB2 or Oracle.
- Have a variety of access methods, although Hierarchical Direct (HDAM) and Hierarchical Indexed Direct (HIDAM) dominate.
- Store data using VSAM, a native z/OS access method, or Overflow Sequential (OSAM), an IMS-specific access method
- Provide an hierarchically structured database, that can be accessed by record or sequentially, and by other sequences that were planned and provided for when the database was designed.

“Fast Path” IMS databases

- Fast Path database types:
 - Data Entry Databases (DEDBs) - similar to HDAM, but optimized
 - Main Storage Databases (MSDBs) - all data stored in memory (I.e. hence can not be too large)
- Neither provide any indexation.
- Instead they are optimized for extremely high transaction rates.
- Fast path DEDBs can only be built atop VSAM.
- DEDBs are particularly suited for use where large databases, or very low processing costs are required, or particularly high data availability, or very high performance is required

“High Availability Large Databases” IMS databases

An extension of IMS full function databases to provide improved:

- availability
- handling of extremely large data volumes
- online reorganization to support continuous availability.

IMS Control Blocks

- When you create an IMS database, you must define the database structure and how the data can be accessed and used by application programs.
- These specifications are defined within the parameters provided in two control blocks, also called DL/I control blocks:
 - Database description (DBD) - describes the physical structure of the database
 - Program specification block (PSB) - describes the database as it will be seen by a particular application program. The PSB tells the application which parts of the database it can access and the functions it can perform on the data.

DL/I

- Application programs use DL/I calls to request data.
 - DL/I is used in batch and online programs to access data stored in databases.
 - DL/I is a command-level language, not a database management system.
 - DL/I then uses system access methods, such as Virtual Storage Access Method (VSAM), to handle the physical transfer of data to and from the database.
 - In Cobol, DL/I is accessed via the statement:

CALL CBLTDLI

DL/I call includes four important parts:

- **FUNCTION:** Tells DL/I what needs to be done (for example, retrieve, update, insert, delete).
- **PCB (program communication block):** Gives the name of the PCB of the database that will be referenced by the call. The database PCB tells IMS which database to use and provides a place for IMS to return results of the call to the application
- **I/O AREA:** Specifies the work area for the requested segment (where the resultant records are written)
- **SSA (Segment search argument):** Specifies the data to be accessed (ex. A specific course), equivalent to the “where” in a SQL statement

DL/I FUNCTION: Tells DL/I what needs to be done

- Functions include:
 - Delete (DEQ)
 - Get Hold Next (GHN)
 - Get Hold Next in Parent (GHNP)
 - Get Hold Unique (GHU)
 - Get Next (GN)
 - Get Next in Parent (GNP)
 - Get Unique (GU)
 - Insert (ISRT)
 - Replace (REPL)
- GU / GHU is used to directly retrieve segments or establish a starting position for sequential processing (ex. GN/GHN)

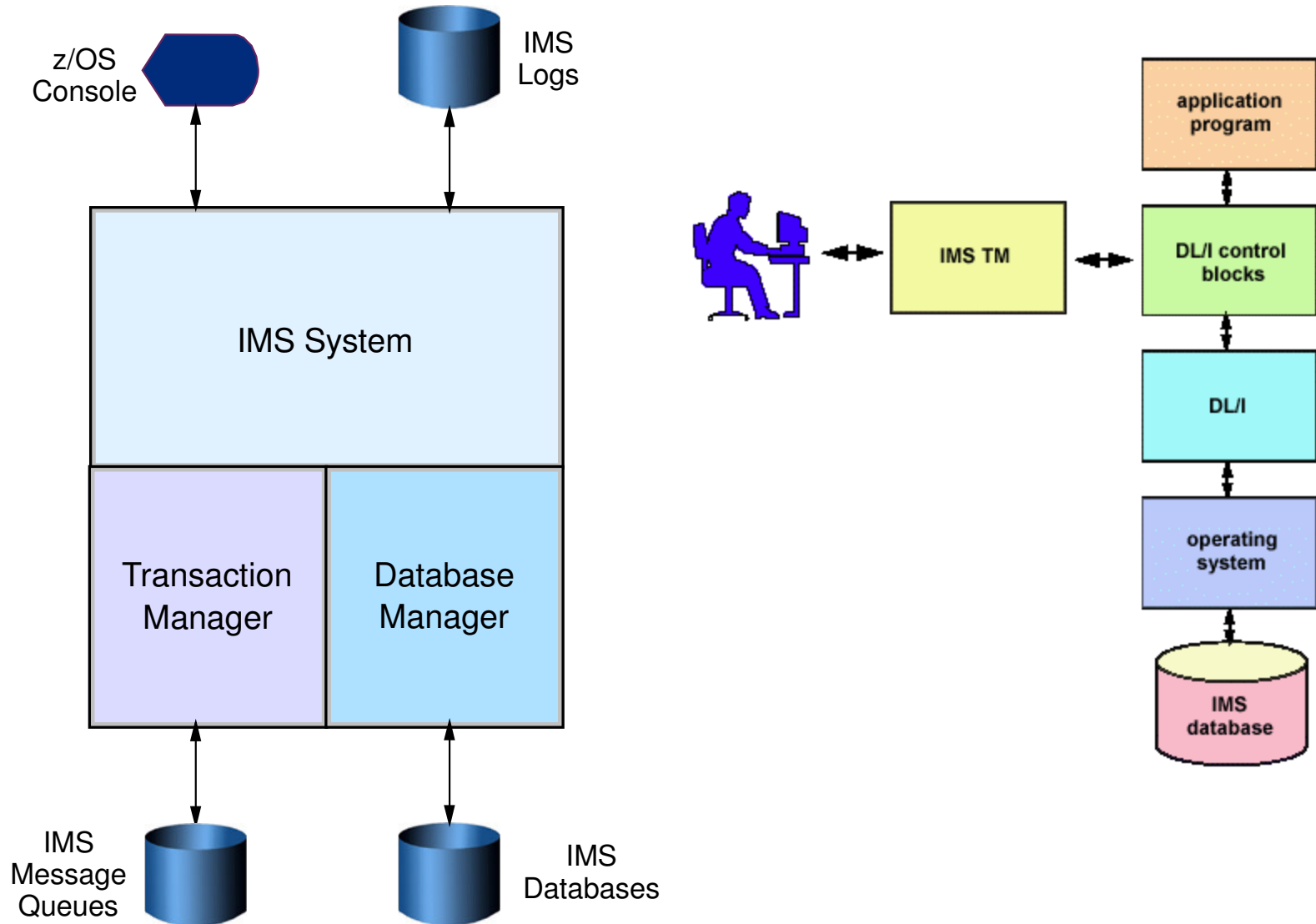
Hold Functions:

Indicates that the application might change the segment (via DLET or REPL)

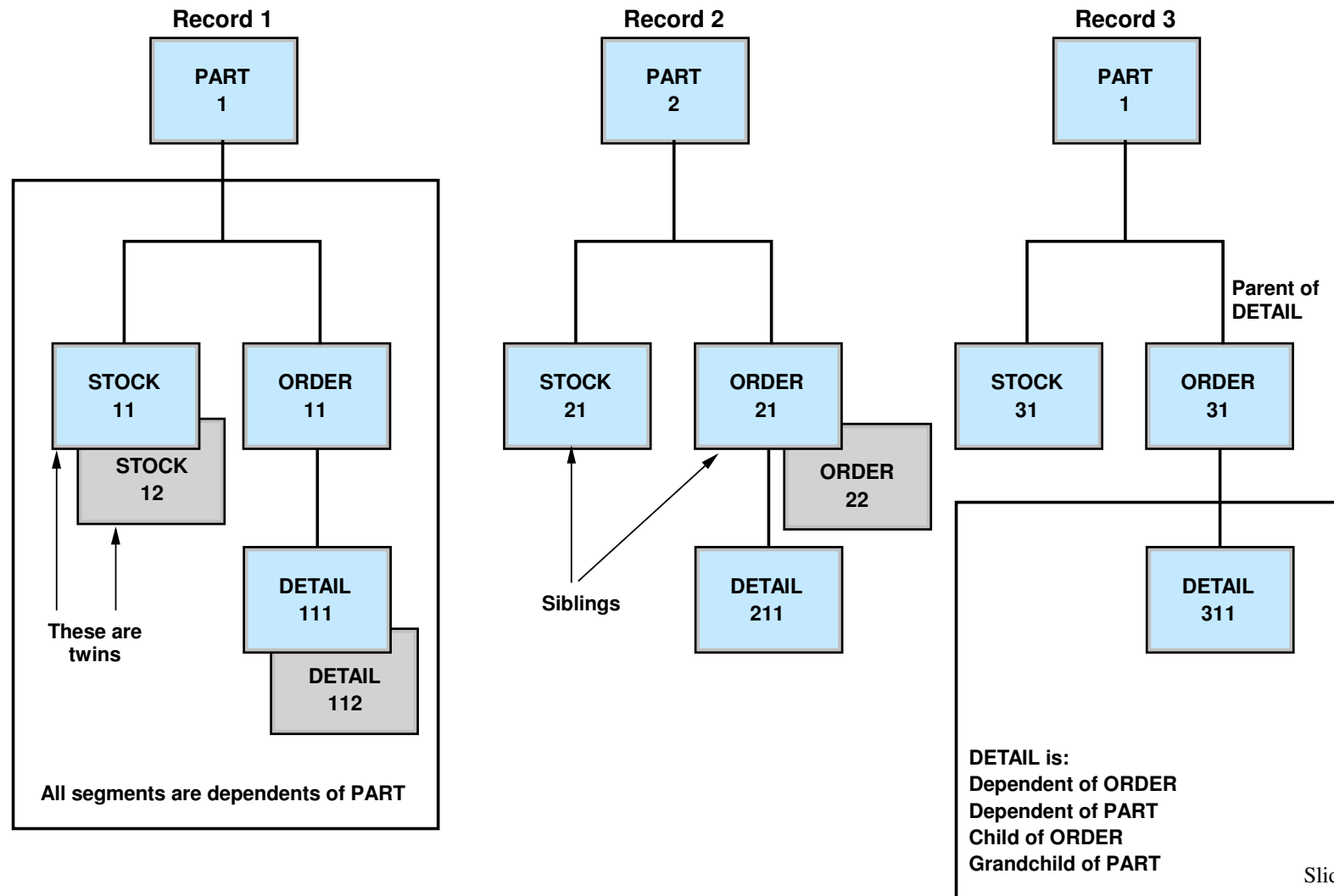
Hold does not force a change

Hold is released when the position in the DB is changed

IMS overview



Segment types and their relationships



Relational Databases

